CPSC 1155 – Assignment 1

Sequential Programming

Objectives

The goal of this assignment is to develop sequential pseudocodes and C++ programs.

Readings

You should be reading Chapters 1 and 2 of the textbook. The lectures and labs will provide additional supporting material.

Instructions

For each of the following problem statements:

1. Read the problem statement and clarify the problem.
2. Determine input, process, and output (IPO).
3. Break the problem into smaller problems if needed.
4. Work out the problem by hand using typical values.
5. Declare the variables and constants (data type + meaningful names).
6. Write a pseudocode as required.
7. Write a C++ program (use the given filename) that implements the pseudocode (or algorithm). Add comments where needed.
8. Make sure to use a comments header to reflect the intention of your program and name of the author (you) and the date the program was written.
9. Test, debug, and execute the program using typical values.

Submit according to the instruction in the " Submission" section.

Problem Statements

1. (volume_sphere.cpp) Write a **C++ program** that calculates and displays the volume of a sphere. The program should prompt the user to enter the radius of the sphere. Assume the input is positive. Here is a sample run:

   ```
   Enter radius of sphere: 1.5
   The volume of sphere is 14.14
   ```

   Make sure to display the output with 2 decimal places. Please check how to use M_PI and pow.

2. (sum_of_digits.cpp) Write a **C++ program** that calculates the sum of digits in a given integer. The program should prompt the user to enter an integer between 0 and 1000 (For now you do not have to test this!). For example, if an integer is 932, the sum of digits is 14.

   [Hint: Use the % operator to extract digits and use / operator to remove the extracted digits.]

Here is a sample run:

```
Enter a number between 0 and 1000: 999
The sum of the digits is 27
```

3. (swimming_pool.cpp)[i] Write a **pseudocode** and a **C++ program** that solves the following problem. Two large and 1 small pump can fill a swimming pool in 4 hours. One large and 3 small pumps can also fill the same swimming pool in 4 hours. The user will enter how many large pumps and small pumps are used. Display how many hours it will take to fill the swimming pool with the specified numbers of pumps. Assume user inputs are larger or equal to 0 and are within reasonable ranges. Here is a sample run:

   ```
   Enter the number of large pumps:4
   Enter the number of small pumps:4
   It will take 1.66667 hours to fill up the pool with 4 large pumps and 4 small
   pumps.
   ```

   Hint: This problem requires you to find the rate of small pump and rate of large pump first. Then we can calculate the hours. See end note i for the original problem and starting equations. You will need to deduct the equations for finding the rate yourself first before programming this problem.

4. (current_time.cpp) The time(0) function in the ctime header file, returns the current time in seconds elapsed since the time 00:00:00 on January 1, 1970 GMT. This time is known as the *UNIX epoch*. You can use this function to obtain the current time, and then compute the current second, minute, and hour. Please do your own calculations instead of using pre-defined members of time_t.

   After accomplishing the above, prompts the user to enter the time zone offset to GMT and displays the time in the specified time zone. Assume input is correct.

   Write a **C++ program** that obtains the current time using time(0) and calculates and displays the current second, minute, and hour.

   Here is a sample run:

   ```
   Current time is 17:40:34 GMT
   Enter the time zone offset to GMT: -5
   The current time in your time zone is 12:40:34
   ```

5. (invest.cpp)[ii] Suppose you find a way to save an additional x dollars per month. Instead of saving it in your bank account, you open a portfolio and invest the money. For simplicity, assume that the annual return is 9% and the principal investment is $5000.

   Write a **pseudocode** and a **C++ program** that prompts the user to enter a monthly contribution amount and number of years and then displays the portfolio value.

   ```
   Enter the monthly contribution amount: 100
   Enter the number of years: 10
   After 10 years, the portfolio value is $31608.21
   ```

   This problem uses the compound interest formula: $A = P(1 + \frac{r}{n})^{nt}$ where A is final amount, P is initial principal balance, r is annual interest rate, n is number of times interest applied per time period t, and t is number of time periods elapsed.

This problem will also use the future value of a series formula: $PMT * (\frac{(1+\frac{r}{n})^{nt}-1}{\frac{r}{n}})$ where PMT is the monthly investment, r is annual interest rate, n is number of times interest applied per time period t, and t is number of time periods elapsed.

Remember to use `setprecision(n)` and `fixed()` for the final value.

6. (wind_chill_temperature.cpp) How cold is outside? The temperature alone is not enough to provide the answer. Other factors including wind speed, relative humidity, and sunshine play important roles in determining coldness outside. In 2001, the National Weather Service (NWS) implemented the new wind-chill temperature to measure the coldness using temperature and wind speed. The formula is:

$$t_{wc} = 35.74 + 0.6215t_a - 35.75v^{0.16} + 0.4275t_a v^{0.16}$$

where t_a is the outside temperature measured in degrees Fahrenheit and v is the speed measured in miles per hour. t_{wc} is the wind-chill temperature. The formula cannot be used for wind speeds below 2 mph or temperatures below –58°F or above 41°F (For now you do not have to test this!).

Write a **pseudocode** and a **C++ program** that prompts the user to enter a temperature between –58°F and 41°F and a wind speed greater than or equal to 2 and displays the wind-chill temperature. Use pow(a, b) to compute $v^{0.16}$.

Here is a sample run:

```
Enter the temperature in Fahrenheit (must be between -58°F and 41°F): 5.3
Enter the wind speed miles per hour (must be greater than or equal to 2): 6
The wind chill index is -5.56707
```

7. (velocity_of_arrow.cpp)[iii] An archer pulls back *x* m on a bow which has a stiffness of *k* N/m. The arrow has mass *m* gram. What is the velocity of the arrow immediately after release?

Hint: Solve this problem by equating the potential energy of the bow to the kinetic energy of the arrow.

$$potential\ energy\ of\ bow = \frac{1}{2}kx^2$$

Where *k* is stiffness and *x* is amount of the bow that is stretched.

Kinetic energy of a particle is:

$$kinetic\ energy\ of\ arrow = \frac{1}{2}mv^2$$

Where *m* is mass and *v* is velocity.

Write a **C++ program** that prompts the user to enter input for pull back distance, stiffness and mass, and displays the velocity of arrow right after release. Assume user inputs are correct.

Here is a sample run:

```
Enter the pull back distance of bow in m: 0.75
Enter the stiffness of bow in N/m: 200
Enter the arrow's mass in g: 50
The velocity of arrow right after release is 47.4 m/s
```

Hint: Remember to convert some units to keep it consistent, and use `setprecision(n)` and `fixed()` to print 1 decimal place.

8. (complex_number.cpp) We can multiply two complex numbers by using the following formula:

$$product\ of\ complex\ number = (x + yi)(u + vi)$$

Where $(x + yi)$ is the first complex number and $(u + vi)$ is the second imaginary number. We can simplify the formula by multiplying the terms and simplifying them.

Write a **C++ program** that prompts the user to enter two complex numbers (the first number is *x* and the second number is *y*) and calculates the product of them.

Hint: $i^2 = -1$. See here for an example and formula.

Here is a sample run:

```
Enter the first complex number (real number followed by imaginary number): 3 2
Enter the second complex number (real number followed by imaginary number): 1
4
The product of 3+2i and 1+4i is -5+14i
```

The inputs can have decimal numbers.

Check showpos and noshowpos to help you print the + sign for imaginary number. Here is an example of how to print 3+2i:

```
cout << noshowpos << 3 << showpos << 2 << "i";
```

Submission

Submit a zip folder named as yourName_Assign1.zip to Brightspace. This folder should consist the **C++ codes** in individual .cpp files and one pseudocode.txt file with all your **pseudocode** (for question 3, 5, and 6). See the document about coding styles on Brightspace for example folder structure.

Please make sure that all your .cpp files compile and run properly before submission. Your file must run properly in order to receive full marks.

Marking Scheme

There are 10 marks for each question with the following details:

Question 1: volume_sphere.cpp

- 5 for correct solution
- 4 for usage of pow and M_PI
- 1 for output with 2 decimal places

Question 2: sum_of_digits.cpp

- must use both % and / operator for full marks

Question 3: swimming_pool.cpp

- 5 for pseudocode

- 5 for correct solution

Question 4: current_time.cpp

- 5 for calculating current time (must do your own calculations)
- 5 for displaying current time after offset

Question 5: invest.cpp

- 5 for pseudocode
- 4 for correct solution
- 1 for output with 2 decimal places

Question 6: wind_chill_temperature.cpp

- 5 for pseudocode
- 5 for correct solution

Question 7: velocity_of_arrow.cpp

- 8 for correct solution
- 2 for output with 1 decimal place

Question 8: complex_number.cpp

- 8 for correct solution
- 2 for output with correct +/- signs using showpos and noshowpos

---

[i] Adapted from Q1 in this math problems set
[ii] Adapted from this example about compound interest formula in the compound interest formula with regular contributions section
[iii] Adapted from Q9 in this physics problems set